J Grid Computing (2019) 17:169–189
https://doi.org/10.1007/s10723-018-9437-3

A Lightweight Service Placement Approach for Community Network Micro-Clouds

Mennan Selimi · Llorenç Cerdà-Alabern · Felix Freitag · Luís Veiga · Arjuna Sathiaseelan · Jon Crowcroft

Received: 20 July 2017 / Accepted: 12 February 2018 / Published online: 28 February 2018

Abstract Community networks (CNs) have gained momentum in the last few years with the increasing number of spontaneously deployed WiFi hotspots and home networks. These networks, owned and managed by volunteers, offer various services to their members and to the public. While Internet access is the most popular service, the provision of services of local interest within the network is enabled by the emerging technology of CN micro-clouds. By putting services closer to users, micro-clouds pursue not only a better service performance, but also a low entry barrier for the deployment of mainstream Internet services within the CN. Unfortunately, the provisioning of these services is not so simple. Due to the large and irregular topology, high software and hardware diversity of CNs, a "careful" placement of micro-clouds services over the network is required to optimize service performance. This paper proposes to leverage state information about the network to inform service placement decisions, and to do so through a fast heuristic algorithm, which is critical to quickly react to changing conditions. To evaluate its performance, we compare our heuristic with one based on random placement in Guifi.net, the biggest CN worldwide. Our experimental results show that our heuristic consistently outperforms random placement by 2x in bandwidth gain. We quantify the benefits of our heuristic on a real live video-streaming service, and demonstrate that video chunk losses decrease significantly, attaining a 37% decrease in the packet loss rate. Further, using a popular Web 2.0 service, we demonstrate that the client response times decrease up to an order of magnitude when using our heuristic. Since these improvements translate in the QoE (Quality of Experience) perceived by the user, our results are relevant for contributing to higher QoE, a crucial parameter for using services from volunteer-based systems and adapting CN micro-clouds as an eco-system for service deployment.

M. Selimi (✉) · A. Sathiaseelan · J. Crowcroft
University of Cambridge, Cambridge, UK
e-mail: mennan.selimi@cl.cam.ac.uk

A. Sathiaseelan
e-mail: arjuna.sathiaseelan@cl.cam.ac.uk

J. Crowcroft
e-mail: jon.crowcroft@cl.cam.ac.uk

L. Cerdà-Alabern · F. Freitag
Universitat Politècnica de Catalunya, BarcelonaTech, Barcelona, Spain

L. Cerdà-Alabern
e-mail: llorenc@ac.upc.edu

F. Freitag
e-mail: felix@ac.upc.edu

L. Veiga
Instituto Superior Técnico (IST), INESC-ID Lisboa, Lisbon, Portugal
e-mail: luis.veiga@inesc-id.pt

Keywords Service placement · Community networks · Micro-clouds · Edge-clouds · Wireless mesh networks

1 Introduction

Since early 2000s, community networks (CNs) or "*Do-It-Yourself*" networks have gained momentum in response to the growing demands for network connectivity in rural and urban communities. The main singularity of CNs is that they are built "bottom-up", mixing wireless and wired links, with communities of citizens building, operating and managing the network. The result of this open, agglomerative, organic process is a very heterogeneous network, with self-managing links and devices. For instance, devices are typically "low-tech", built entirely by off-the-shelf hardware and open source software, which communicate over wireless links. This poses several challenges, such as the lack of service guarantees, inefficient use of the available resources, and absence of security, to name just a few.

These challenges have not precluded CNs from flourishing around. For instance, Guifi.net,[1] located in the Catalonia region of Spain, is a successful example of this paradigm.

Guifi.net is a "crowdsourced network", i.e., a network infrastructure built by citizens and organizations who pool their resources and coordinate their efforts to make these networks happen [7]. In this network, the infrastructure is established by the participants and is managed as a common resource [5]. Guifi.net is the largest and fast growing CN worldwide. Some measurable indicators are the number of nodes (> 34,000), the geographic scope (> 50,000 km of links), the Internet traffic etc. Regarding the Internet traffic, Fig. 1 depicts the evolution of the total inbound (i.e., pink color) and outbound (i.e., yellow color) traffic from and to the Internet for the last two years. A mere inspection of this figure tells us that Guifi.net traffic has tripled (i.e., 3 Gbps peak). Traffic peaks correspond to the arrival of new users and deployment of bandwidth-hungry services in the network. Actually, a significant number of services, including GuifiTV, graph servers, mail and game services, are running within Guifi.net. All these services have been provided by individuals, social groups, and small non-profit or commercial service providers.

Guifi.net ultimate aim is to create a full digital ecosystem that covers a highly localized area. But this mission is not so simple. A quick glance at the type of services that users demand reveals that the percentage of the Internet services (e.g., proxies) is higher than 50% [13, 30]. This confirms that Guifi.net users are typically interested in mainstream Internet services, which imposes a heavy burden on the "'thin" backbone links, with users experiencing high service variability. The main reasons why the local services have not been developed within CNs or have not gained traction among the members, is the lack of streamlined mechanisms to exploit all the resources available within the CNs. As a result, the development of these types of services can be very challenging.

The current network deployment model in the Guifi.net CN is based on geographic singularities rather than on the QoS (Quality of Service). The resources in the network are not uniformly distributed [41]. Wireless links are with asymmetric quality for the services and there is a highly skewed traffic and bandwidth distribution [10].

Further, the network topology in a wireless CN such as Guifi.net is organic and different with respect to conventional ISP (Internet Service Provider) networks [44]. Guifi.net is composed of numerous distributed CNs and they represent different types of network topologies. The overall topology is constantly changing and there is no fixed topology as in the Data Center (DC) environment. The Guifi.net network shows some typical patterns from the urban networks (i.e., mesh networks) combined with an unusual deployment, that do not completely fit neither with organically grown networks nor with planned networks [42]. This implies that a service placement solution (i.e., algorithm) that works in a certain topology might not work in another one.

The infrastructure in the Guifi.net CN is highly unreliable and heterogeneous [41]. Devices and the network are very heterogeneous compared to the DCs where they are very homogeneous. The strong heterogeneity is due to the diverse capacity of nodes and links, as well as the asymmetric quality of wireless links. Employed technologies in the network vary significantly, ranging from very low-cost, off-the-shelf wireless (WiFi) routers, home gateways, laptops to expensive optical fiber equipment [4, 32]. In terms of

[1] http://guifi.net/

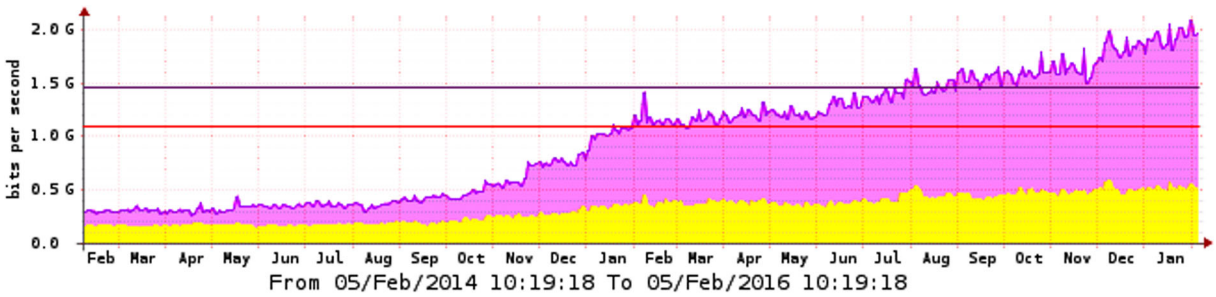


Fig. 1 Guifi.net inbound and outbound traffic (2014–2016)

demand distribution, the demand comes directly from the edge so there are no central load balancers as in the DC environments.

Among other issues, the above-mentioned challenges spurred the invention of "alternative" service deployment models to cater for users in the Guifi.net. One of these models was that based on *micro-clouds*. A micro-cloud is nothing but a platform to deliver services to a local community of citizens within the vast CN. Services can be of any type, ranging from personal storage [29] to video streaming and P2P-TV [28]. Observe that this model is different from Fog computing [9, 21], which extends cloud computing by introducing an intermediate layer between devices and datacenters. Micro-clouds take the opposite track, by putting services closer to consumers, so that no further or minimal action takes place in the Internet. The idea is to tap into the shorter, faster connectivity between users to deliver a better service and alleviate overload in the backbone links.

This approach, however, poses new challenges, such as that of *the optimal placement of micro-clouds* within the CN to overcome suboptimal performance. And Guifi.net is not an exception. Obviously, a placement algorithm that is agnostic to the state of the underlying network may lead to important inefficiencies. Although conceptually straightforward, it is challenging to calculate an optimal decision due to the dynamic nature of CNs and usage patterns.

This paper tries to answer the following three research questions:

1. First, given that sufficient state information is in place, is network-aware placement enough to deliver satisfactory performance to CN users?
2. Second, can the redundant placement of services further improve performance?
3. Third, given a CN micro-cloud infrastructure, what is an effective and low-complexity service placement solution that maximizes the end-to-end performance (e.g., bandwidth), taking into account the dynamic behavior of the network and resource availability?

To answer these questions, we contribute in this work with a new placement heuristic called *BASP* (Bandwidth and Availability-aware Service Placement), which uses the state of the underlying CN to optimize service deployment [27]. In particular, it considers two sources of information: i) network bandwidth and ii) node availability to make optimized decisions. Compared with brute-force search, which takes in the order of hours to complete, BASP runs much faster; it just takes a few seconds, while achieving equally good results.

Our results show that the BASP heuristic consistently outperforms random placement, the existing in-place and naturally fast strategy in Guifi.net, by $2x$ with respect to end-to-end bandwidth gain. Driven by these findings, we then ran BASP in a real CN and quantified the boost in performance achieved after deploying a live video-streaming and Web 2.0 service according to BASP. Our experimental results demonstrate that with BASP, the video chunk loss in the peer side decreased up to a 3% point reduction, i.e., worth a 37% reduction in the packet loss rate, which is a significant improvement. Furthermore, when using the BASP with the Web 2.0 service (i.e., social networking service), the client response times decreased up to an order of magnitude.

The rest of the paper is organized as follows. In Section 2 we define CN micro-clouds and describe and characterize the performance of the production CN such as QMP (Quick Mesh Project) network, which

is a subset of the Guifi.net CN. Section 3 defines our system model and presents our BASP heuristic. In Section 4 we discuss the evaluation results of our BASP heuristic, using the QMP network traces. In Section 5 we present and discuss the real deployment experiments with a video-streaming and Web 2.0 service. Section 6 describes related work and Section 7 concludes and discusses future research directions.

2 Background and Network Characterization

The adoption of the CN micro-cloud services requires carefully addressing the service deployment and performance requirements. Our service placement strategy considers two aspects: node availability and network bandwidth. As the first step, it is vital to understand the behavior of these two dimensions in a real CN. We achieve this by characterizing over a five-month period a production wireless CN such as the QMP network, which is a subset of the Guifi.net. Our goal is to determine the key features of the network (e.g. bandwidth, traffic distribution), of the nodes (e.g., availability patterns) and service types in the network that could help us to design new heuristics for intelligent service placement in CNs.

2.1 Micro-Clouds in the Community Networks

CN micro-clouds are built on top of the CNs. In this model, a cloud is deployed closer to CN users and other existing network infrastructure (e.g., public schools, strategic locations etc.). CN micro-clouds take the opposite track from Fog Computing, by putting services closer to consumers, so that no further or minimal action takes place in the Internet. In CN micro-clouds, by contrast to other edge computing models, the users of edge services are enabled to collaborate and actively participate in the service provision, and contribute to sustain edge micro-clouds. They are deployed over a single or set of user nodes, and comparing to the public clouds they have a smaller scale, so one still gets high performance due to locality and control over service placement.

The devices forming the CN micro-clouds are co-located in either users homes (e.g., as home gateways, routers, laptops, parabolic antennas etc., as shown in the Fig. 2) or distributed in the CNs. The concept of micro-clouds can also be introduced in order to split deployed CN nodes into different groups. For instance, a micro-cloud can refer to these nodes which are within the same service announcement and discovery domain. Different criteria can be applied to determine to which micro-cloud a node belongs to. Applying technical criteria (e.g., Round-trip time (RTT), bandwidth, number of hops, resource characteristics) for micro-cloud assignment is a possibility to optimize the performance of several services. But also social criteria may be used, e.g., bringing in a micro-cloud cloud resources together from users which are socially close may improve acceptance, the willingness to share resources and to maintain the infrastructure.

Fig. 2 Devices forming a CN micro-cloud (home gateways, routers, laptops, set-top boxes, antennas etc.)

2.2 The QMP Network: an Urban CN of the Guifi.net

QMP network began to operate in 2009 in a quarter of the city of Barcelona, Spain, called Sants, as part of the Quick Mesh Project (QMP).[2] The QMP network is an urban mesh network and it is a subset of the Guifi.net CN sometimes called GuifiSants. At the time of writing, the QMP has around 77 nodes. There are two gateways (i.e., proxies) distributed in the network that connect the QMP to the rest of Guifi.net and the Internet (highlighted in the Fig. 3). A detailed description of QMP can be found in [10].

Typically, the QMP users have an outdoor router (OR) with a Wi-fi interface on the roof, connected through Ethernet to an indoor AP (access point) as a premises network. The most common OR in the QMP is the NanoStation M5 as shown in the Fig. 2,

[2] http://qmp.cat

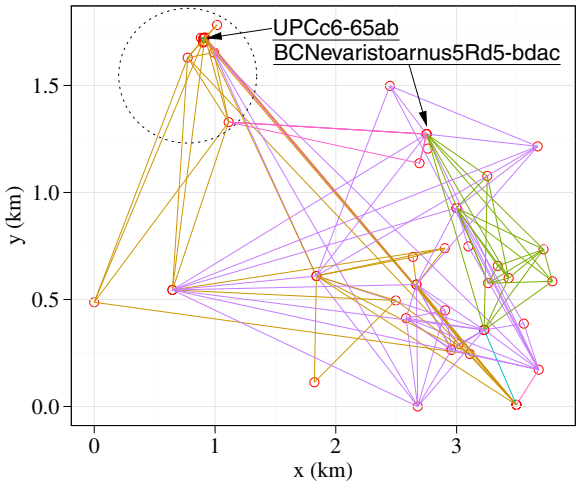


Fig. 3 QMP network topology

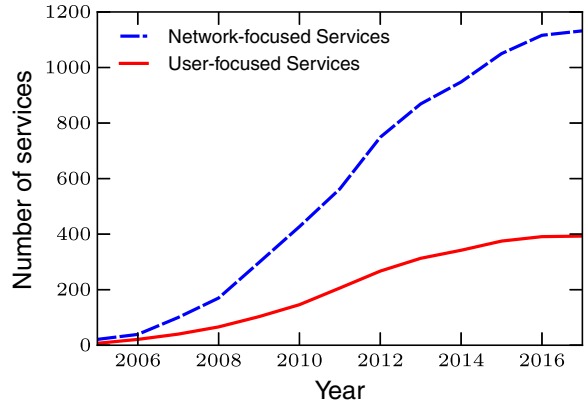


Fig. 4 Number of local services in the Guifi.net (network and user-focused)

which is used to build point-to-point links in the network and integrates a sectorial antenna with a router furnished with a wireless 802.11an interface. Some strategic locations have several NanoStations, that provide larger coverage. In addition, some links of several kilometers are set up with parabolic antennas (NanoBridges). ORs in the QMP are flashed with the Linux distribution which was developed inside the QMP project which is a branch of OpenWRT[3] and uses *BMX6* and *BMX7* as the routing protocol [25].

The user devices connected to the ORs consists of Minix Neo Z64 and Jetway mini PCs, which are equipped with an Intel Atom CPU. They run the Cloudy[4] operating system, which leverages the Docker containerization technology and allows CN users to launch their favorite or the predefined Docker images in a few clicks, from their browser. This rapid application provision allows room for new, very dynamic ways to deploy services and share resources in a digital community.

Methodology and Data Collection The measurements have been obtained by connecting via SSH to each QMP OR and running basic system commands available in the QMP distribution. This method has the advantage that no additional software needs to be installed on the nodes. Live measurements have been taken hourly over a five-month period, starting from July 2016 to November 2016, and our live monitoring page and data is publicly available in the Internet.[5] We use this data to analyze the main aspects of the QMP network.

2.3 Services in the QMP Network

In the Guifi.net (QMP) CN, the Internet cloud services have equivalent alternatives that are owned and operated at the community level. There are two type of services in the network: network-focused and user-focused services. Figure 4 depicts the evolution of user and network-focused services during the last 10 years. Considering that network management is of interest to all users in the network (i.e., to keep the network up and running), Fig. 4 reveals that services related to the network operation outnumber the local services intended for end-users. However, in the recent years the local user services are also gaining attraction as demonstrated by the Fig. 4.

Moreover, the most frequent of all the services, whether user-focused or network-focused, are the proxy services [12]. Proxies act as free gateways to the Internet for the CN users. Specifically for the user-focused services, the percentage of the Internet access services (i.e., proxies and tunnel-based) is higher than 55%, confirming that the users of Guifi.net are typically interested in accessing the Internet [30]. Further, other important services are web hosting, data storage, VoIP, and video streaming. From the service placement point of view, we are focusing on both type of services in the network.

[3] https://openwrt.org/

[4] http://cloudy.community/

[5] http://dsg.ac.upc.edu/qmpsu/

2.4 Node Availability

The quality and state of the heterogeneous hardware used in the QMP influences the stability of the links and network performance. Availability of the QMP nodes is used as an indirect metric for the quality of connectivity that new members expect from the network.

Figure 5 shows the Empirical Cumulative Distribution Function (ECDF) of the node availability collected for a period of five months. We define the availability of a node as the percentage of times that the node appears in a *capture*, counted since the node shows up for the first time. A capture is an hourly network snapshot that we take from the QMP network (i.e., we took 2718 captures in total). Figure 5 reveals that 25% of the nodes have an availability lower than 90% and others nodes left have an availability between 90–100%. In a CN such as QMP, users do not tend to deliberately reboot the device unless they have to perform an upgrade, which is not very common. Hence, the percentage of times that node appears in a capture is a relatively good measure of the node availability due to random failures.

When we compare the availability distribution reported in a similar study and environment on PlanetLab [43], a QMP node has a higher probability of being disconnected or not to be reachable from the network. The fact that PlanetLab showed a higher average availability (i.e., sysUpTime) on its nodes may be because it is an experimental testbed running on much more stable computers and environment. Furthermore, the QMP members are not only responsible for the maintenance of their nodes, but also for ensuring a minimum standard of connectivity with other parts of the network.

Figure 6 depicts the number of nodes and links during captures. Figure shows that the QMP is growing. Overall, 77 different nodes were detected. From those, 71 were alive during the entire measurement period. Around 6 nodes were missed in the majority of the captures. These are temporarily working nodes from other mesh networks and laboratory devices used for various experiments. Figure 6 also reveals that on average 175 of the links used between nodes are bidirectional and 34 are unidirectional. For bidirectional links, we count both links in opposite direction as a single link.

In summary, node availability is important to identify those nodes that will minimize service interruptions over time. Based on the measurements, we assign availability scores (R_n) to each of the nodes. The highly available nodes are the possible candidates for deploying on them the micro-cloud services.

2.5 Bandwidth Characterization

A significant amount of services that run on the QMP and Guifi.net network are network-intensive (i.e., bandwidth and delay sensitive), transferring large amounts of data between the network nodes [8, 30]. The performance of such kind of services depends not

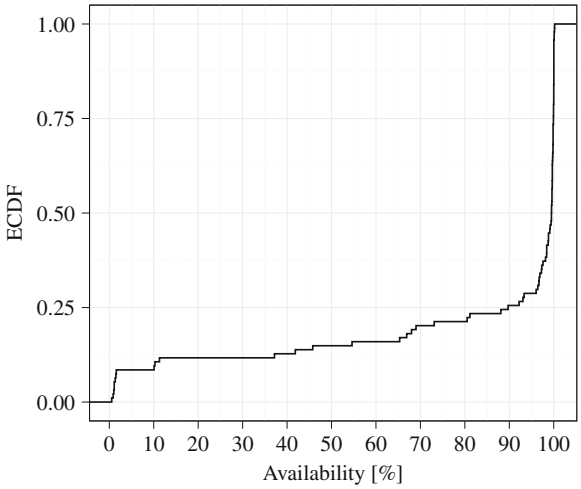


Fig. 5 Node availability in the QMP network

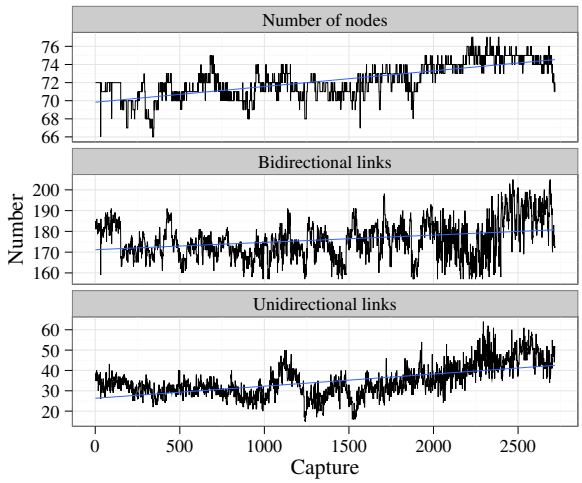


Fig. 6 Node and link presence in the QMP network

just on computational and disk resources but also on the network bandwidth between the nodes on which they are deployed. Therefore, considering the network bandwidth when placing services in the network is of high importance.

First, we characterize the wireless links of the QMP network by studying their bandwidth. Figure 7 shows the average bandwidth distribution of all the links. The figure shows that the link throughput can be fitted with a mean of 21.8 Mbps. At the same time Fig. 7 reveals that the 60% of the nodes have 10 Mbps or less throughput. The average bandwidth of 21.8 Mbps obtained in the network allows many popular bandwidth-hungry service to run without big interruptions. This high performance can be attributed to the 802.11an devices used in the network.

In order to see the variability of the bandwidth, Fig. 8 shows the bandwidth averages in both directions of the three busiest links. Upload operation is depicted with a solid line and download operation with a dashed line. The nodes of three busiest links are highlighted on the top of the figure. We noted that the asymmetry of the bandwidths measured in both directions it not always due to the asymmetry of the user traffic (not shown in the graphs). For instance, node GSgranVia255, around 6 am, when the user traffic is the lowest and equal in both directions, the asymmetry of the links bandwidth observed in Fig. 8 remains the same. We thus conclude that even though bandwidth time to time is slightly affected by the traffic, the asymmetry of the links that we see might be due to the

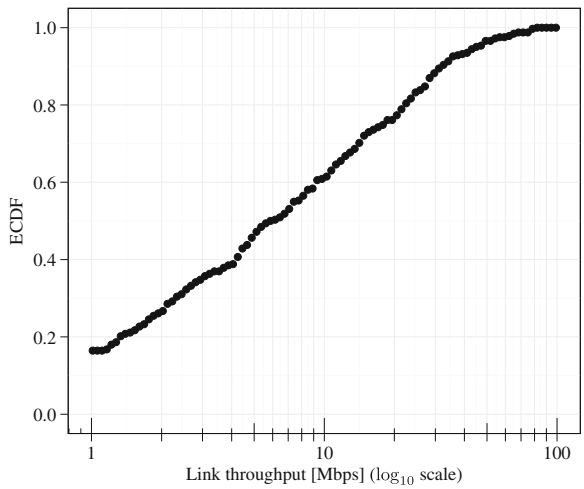


Fig. 7 Bandwidth distribution of the links

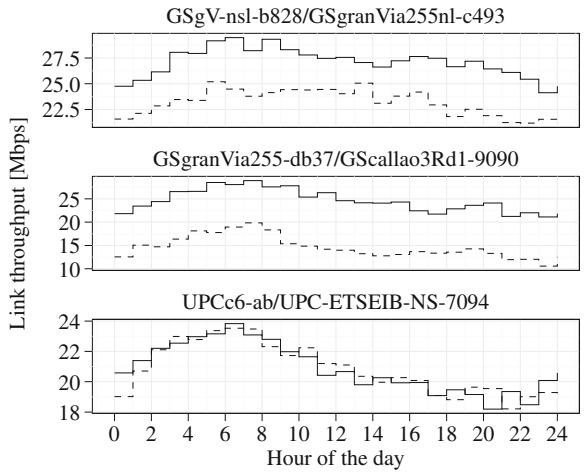


Fig. 8 Bandwidth in three busiest links

link characteristics, as level of interferences present at each end, or different transmission powers.

In order to measure the link asymmetry, Fig. 9 depicts the bandwidth measured in each direction. A boxplot of the absolute value of the deviation over the mean is also depicted on the right. The figure shows that around 25% of the links have a deviation higher than 40%. At the same time, the other 25% of the links have a deviation less than 10%. After performing some measurements regarding the signaling power of the devices, we discovered that some of the community members have re-tuned the radios of their devices (e.g., transmission power, channel and other parameters), trying to achieve better performance, thus, changing the characteristics of the links. Thus, we can conclude that the symmetry of the links, an assumption often used in the literature of wireless mesh networks, is not very realistic for our case and service placement algorithms unquestionably need to take this into account.

2.6 Discussion

Here are some observations (features) that we have derived from the measurements in the QMP network:

Dynamic Topology The QMP network is highly dynamic and diverse due to many reasons, e.g., its community nature in an urban area; its decentralized organic growth with extensive diversity in the technological choices for hardware, wireless media, link protocols, channels, routing protocols etc.; its

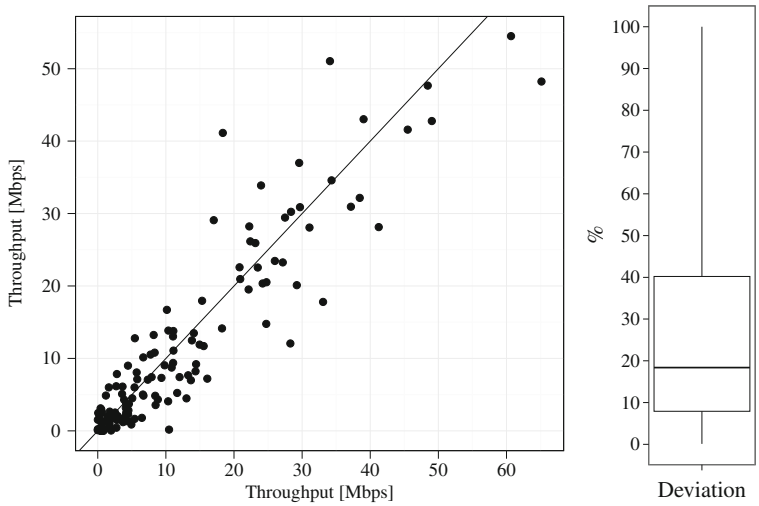


Fig. 9 Bandwidth asymmetry

mesh topology etc. The current network deployment model is based on geographic singularities rather than QoS. The network is not scale-free. The topology is organic and different with respect to conventional ISP networks.

Non-uniform Resource Distribution The resources are not uniformly distributed in the network. Wireless links are with asymmetric quality for services (25% of the links have a deviation higher than 40%). We observed a highly skewed traffic pattern and highly skewed bandwidth distribution (Fig. 7).

Currently used organic (i.e., random) placement scheme in the QMP and Guifi.net in general, is utterly inefficient, failing to capture the dynamics of the network and therefore it fails to deliver the satisfying QoS. The strong assumption under random service placement, i.e., uniform distribution of resources, does not hold in such environments.

Furthermore, the services deployed have different QoS requirements. Services that require intensive inter-component communication (e.g., streaming service), can perform better if the replicas (i.e., service components) are placed close to each other in the high capacity links [28]. On other side, bandwidth-intensive services (e.g., distributed storage, video-on-demand) can perform much better if their replicas are as close as possible to their final users (i.e., overall reduction of bandwidth for service provisioning) [31].

Our goal is to build on this insight and design a network-aware service placement heuristic that will improve the service quality and network performance by optimizing the usage of scarce resources in CNs such as bandwidth.

3 Context and Problem

Based on the network measurements we did at the QMP network, in this section, first we describe our model for network and service graph. Subsequently we build on this to describe the service placement problem. The symbols used in this section are listed in Table 1.

3.1 Network Graph

The deployment and sharing of services in CNs is made available through *community network micro-clouds* (CNMCs). The idea of CNMC is to place the

Table 1 Input variables

Symbol	Description
N	Set of physical nodes in the network
E	Set of edges (physical links) in the network
S	Set of services
D	Set of service copies
k	Max number of service copies
B_e	Bandwidth capacity of link e
$\beta_{s1,s2}$	Bandwidth requirement between services s_1 and s_2
R_n	Availability of node n
λ	Availability threshold

cloud at the edge closer to community end-users, so users can have fast and reliable access to the service. To reach its full potential, a CNMC needs to be carefully deployed in order to effectively take advantage and utilize efficiently the available bandwidth resources.

In a CNMC, a server or low-power device (i.e, home gateway) is directly connected to the wireless base-station (ORs) providing cloud services to users that are either within a reasonable distance or directly connected to the base-station.

We call the CN the *underlay* to distinguish it from the *overlay* network which is built by the services. The underlay network is supposed to be connected and we assume each node knows whether other nodes can be reached (i.e., next hop is known). We can model the underlay graph as: $G \leftarrow (N, E)$ where N is the set of nodes connected to the outdoor routers (ORs) present in the CNs and E is the set of wireless links that connects them. Physical links between nodes are characterized by a given bandwidth (B_i). Furthermore, each link has a bandwidth capacity (B_e) (i.e., theoretical capacity). Each node in the network has an availability score (R_n) derived from the real measurements in the QMP network.

3.2 Service Graph

The services aimed in this work are at infrastructure level (IaaS), as cloud services in current dedicated datacenters. Therefore, the services are deployed directly over the core resources of the network and accessed by the clients. Services can be deployed by QMP users or administrators.

The services we consider in this work are distributed services (i.e., independently deployable services as in the Microservices Architecture[6]). The distributed services can be composite services (non-monolithic) built from simpler parts, e.g., video streaming (built from the source and the peer component), web service (built from the database, the memcached and the client component) etc. In the real deployment, one service component corresponds to one Docker container. These parts or components of the services create an overlay and interact with each other to offer more complex services. Bandwidth requirement between two services s_1 and s_2 is given by $\beta_{s1,s2}$. At most k copies can be placed for each service s.

A service may or may not be tied to a specific node of the network. Each node can host one or more type of services. In this work we assume an offline service placement approach where a single or a set of applications are placed "in one shot" onto the underlying physical network, i.e., different from online placement [45]. We might rearrange (migrate) the placement of the same service over the time because of the service performance fluctuation (e.g., weather conditions, node availability, changes in use pattern, and etc.). We do not consider real-time service migration.

3.3 Service Placement Problem

The concept of service and network graph allows us to formulate the problem statement more precisely as: "*Given a service and network graph, how to place a service on a network as to maximize user QoS and QoE, while satisfying a required level of availability for each node (N) and considering a maximum of k service copies?*"

Let B_{ij} be the bandwidth of the path to go from node i to node j. We want a partition of k clusters (i.e., services) : $C \leftarrow C_1, C_2, C_3, \ldots, C_k$ of the set of nodes in the mesh network. The cluster head i of cluster C_i is the location of the node where the service will be deployed. The partition maximizing the bandwidth from the cluster head to the other nodes in the cluster is given by the objective function:

$$\arg\max_C \sum_{i=1}^{k} \sum_{j \in Ci} B_{ij} \tag{1}$$

with respect to the following constraints:

1. The total bandwidth used per link cannot exceed the total link capacity:

$$\forall e \in \mathbb{E} : \sum_{s1,s_2 \in \mathbb{S}} \beta_{s_1,s_2}(e) \leq B_e \tag{2}$$

2. Availability-awareness: the node availability should be higher than the predefined threshold λ:

$$\forall n \in \mathbb{N} : \sum_{n \in \mathbb{N}} R_n \geq \lambda \tag{3}$$

3. Admission control: At most, k copies can be placed for each service:

$$|D| \leq k \tag{4}$$

[6] http://microservices.io/patterns/

3.4 Proposed Heuristic Algorithm: BASP

Solving the problem stated in the (1) in brute force for any number of N and k is NP-hard and very costly. The naive brute force method can be estimated by calculating the *Stirling number of the second kind* [1] which counts the number of ways to partition a set of n elements into k nonempty subsets, i.e., $\frac{1}{k!}\sum_{j=0}^{k}(-1)^{j-k}\binom{n}{k}j^n \Rightarrow \mathcal{O}(n^k k^n)$. Thus, due to the obvious combinatorial explosion, we propose a low-cost and fast heuristic called BASP.

The BASP (Bandwidth and Availability-aware Service Placement) allocates services taking into account the bandwidth of the network and the node availability. BASP is executed every single time a (new) service deployment is about to be made. In every run, the BASP partitions the network topology into k (maximum allowed number of service replicas) and removes the nodes that are under the pre-defined availability threshold (Phase 1); estimates and computes the bandwidth of the nodes (Phase 2); and finally re-assigns nodes to selected clusters (Phase 3). Algorithm 1 depicts the pseudo-code and Fig. 10 demonstrates the phases of the BASP.

The BASP runs in three phases:

1. **Phase 1: Availability-awareness and K-Means:** Initially in this phase we check the availability of the nodes in the network. The nodes that are under the predefined availability threshhold are removed. Then, we use the naive K-Means partitioning algorithm in order to group nodes based on their geo-location. The idea is to get back clusters of nodes that are close to each other. The K-Means algorithm forms clusters of nodes based on the Euclidean distances between them, where the distance metrics in our case are the geographical coordinates of the nodes. In traditional K-Means algorithm, first, k out of n nodes are randomly selected as the cluster *centroids* depicted with a purple color in Fig. 10 (e.g., nodes E, Z and T). Each of the remaining nodes decides its cluster centroid nearest to it according to the Euclidean distance. After each of the nodes in the network is assigned to one of k clusters, the centroid of each cluster is re-calculated. Each cluster contains a full replica of a service, i.e., the algorithm in this phase partitions the network topology into k (i.e., maximum allowed number of service replicas) clusters. Grouping nodes based on geo-location

Algorithm 1 B A S P

Require: $G(N, E)$ ▷ Network graph (qmpTopology.xml)
`k` ▷ k partition of clusters $\mathtt{C} \leftarrow C_1, C_2, C_3, \ldots, C_k$
`Bi` ▷ bandwidth of the node i
`Rn` ▷ availability of the node n
`λ` ▷ availability threshold

Phase 1 – Availability-awareness and K-Means

1: **procedure** AVAILABILITYAWARENESSKMEANS (G, R_n, k)
2: **if** $R_n \geq \lambda$ **then**
3: $PerformKMeans$(G,k)
4: **return** C
5: **end if**
6: **end procedure**

Phase 2 – Aggregate Bandwidth Maximization

7: **procedure** FINDCLUSTERHEADS(C)
8: $clusterHeads \leftarrow list()$
9: **for all** $k \in C$ **do**
10: **for all** $i \in C_k$ **do**
11: $B_i \leftarrow 0$
12: **for all** $j \in setdiff(C, i)$ **do**
13: $B_i \leftarrow B_i + estimate.route.bandwidth(G, i, j)$
14: **end for**
15: $clusterHeads \leftarrow \max B_i$
16: **end for**
17: **end for**
18: **return** $clusterHeads$
19: **end procedure**

Phase 3 – Cluster Re-Computation

20: **procedure** RECOMPUTECLUSTERS (*clusterHeads, G*)
21: $C\prime \leftarrow list()$
22: **for all** $i \in clusterHeads$ **do**
23: $cluster_i \leftarrow list()$
24: **for all** $j \in setdiff(G, i)$ **do**
25: $B_j \leftarrow estimate.route.bandwidth(G, j, i)$
26: **if** B_j is best from other nodes i **then**
27: $cluster_i \leftarrow j$
28: **end if**
29: $C\prime \leftarrow cluster_i$
30: **end for**
31: **end for**
32: **return** $C\prime$
33: **end procedure**

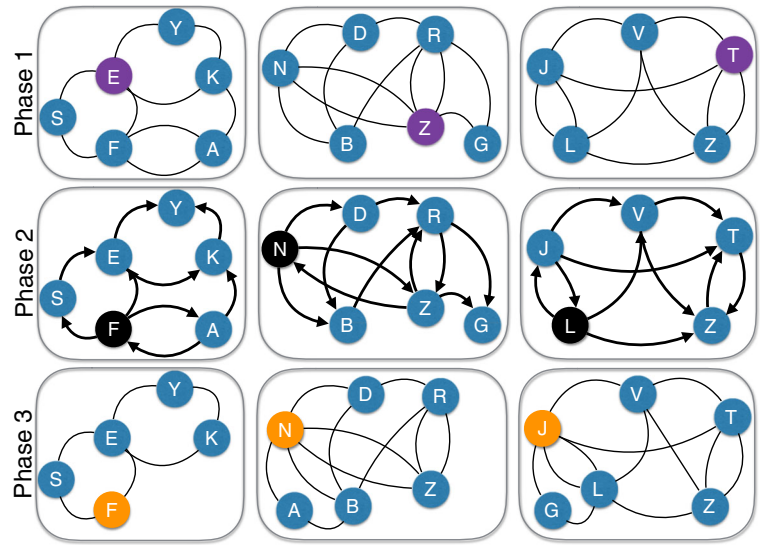


Fig. 10 Phases of the BASP algorithm

is in line with how the QMP is organized. The nodes in the QMP are organized into a tree hierarchy of *zones*. A zone can represent nodes from a neighborhood or a city. Each zone can be further divided in child zones that cover smaller geographical areas where nodes are close to each other. From the service perspective we consider placements inside a particular zone. We use K-Means with geo-coordinates as an initial heuristic for our algorithm. As an alternative, clustering based on network locality can be used. Several graph community detection techniques are available for our environment [20].

2. **Phase 2: Aggregate Bandwidth Maximization:** The second phase of the algorithm is based on the concept of finding the cluster *heads* maximizing the bandwidth between them and their member nodes in the clusters C_k formed in the first phase. The cluster heads computed are depicted with a black color in Fig. 10 (e.g., nodes F, N and L). The bandwidth between two nodes is estimated as the bandwidth of the link having the minimum bandwidth in the shortest path. The cluster heads computed are the candidate nodes for the service placement. This is plotted as Naive K-Means in the Fig. 11.

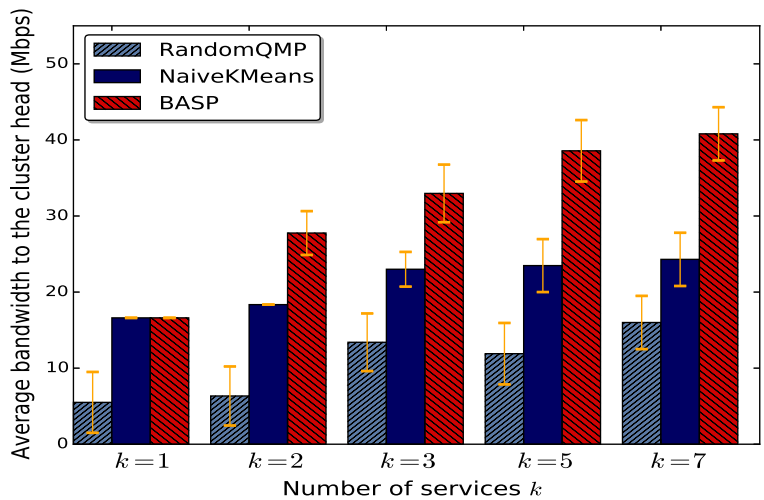


Fig. 11 Average bandwidth to the cluster heads

3. **Phase 3: Cluster Re-Computation:** The third and the last phase of the algorithm includes reassigning the nodes to the selected cluster heads having the maximum bandwidth, since the geolocation of the nodes in the clusters formed during phase one is not always correlated with their bandwidth. The final cluster heads computed are depicted with an orange color in Fig. 10 (e.g., nodes F, N and J) . This way the clusters are formed based on nodes bandwidth. This is plotted as *BASP* in the Fig. 11.

Complexity The complexity of the *BASP* is as follows: for *BASP*, finding the optimal solution to the K-means (i.e., phase one) clustering problem if k and d (the dimension) are fixed (e.g., in our case $n = 77$, and $d = 2$), the problem can be exactly solved in time $\mathcal{O}(n^{dk+1} \log n)$, where n is the number of entities to be clustered. The complexity for computing the cluster heads in phase two is $\mathcal{O}(n^2)$, and $\mathcal{O}(n)$ for the reassigning the clusters in phase three. Therefore, the overall complexity of BASP is polylogarithmic $\mathcal{O}(n^{2k+1} \log n)$, which is significantly smaller than the brute force method and thus practical for commodity processors.

4 Evaluation

Setup We take a network snapshot (i.e., capture) from 77 physical nodes of the QMP network regarding the bandwidth of the links and node availability. The data obtained has been used to build the topology graph of the QMP. The QMP topology graph is constructed by considering only operational nodes, marked in "working" status, and having one or more links pointing to another node. Additionally, we have discarded some disconnected clusters. The links are bidirectional and unidirectional, thus we we use a directed graph. The nodes of QMP consists of Intel Atom N2600 CPU, 4 GB of RAM and 120 GB of disk space. Our experiment is comprised of 5 runs and the presented results are averaged over all the runs. Each run consists of 15 repetitions.

4.1 Comparison

To emphasize the importance of the different phases of the Algorithm 1, we compare in this section two phases of our heuristic algorithm with the *Random Placement*, i.e., the default placement at the QMP.

Random Placement Currently, the service deployment (much as network deployment) at the QMP is not centrally planned but initiated individually by the CN members. Public, user and community-oriented services are placed randomly on super-nodes and users' premises, respectively. The only parameter taken into account when placing services is that the devices must be in "production" state. The network is not taken into consideration at all. All nodes in the production state appear equally to the users.

Naive K-Means Placement This corresponds to the second phase of the heuristic Algorithm 1. The service is placed on the node having the maximum bandwidth on the initial clusters formed by K-Means. We limit the choice of the cluster heads to be inside the sets of clusters obtained using K-Means.

BASP Placement It includes the three phases of the heuristic Algorithm 1. The service is placed on the node having the maximum bandwidth after the clusters are re-computed.

4.2 Results

Figure 11 depicts the average bandwidth to the cluster heads obtained with the *Random*, *Naive K-Means* and the *BASP* heuristic algorithm. This value reflects the average bandwidth computed from the cluster heads obtained, to the other non-cluster nodes within each cluster.

Figure 11 reveals that for the considered number of services k, *BASP* outperforms both *Naive K-Means* and *Random* placement. For $k = 2$, the average bandwidth to the cluster heads has increased from 18.3 Mbps (*Naive K-Means*) to 27.7 Mbps (*BASP*), which represents a 50% improvement. The highest increase of 67% is achieved when $k = 7$. On average, when having up to 7 services in the network, the gain of *BASP* over *Naive K-Means* is of 45%. Based on the observations from Fig. 11, the gap between the two algorithms grows as k increases. We observe that k will increase as the network grows. And hence, *BASP* will presumably render better results for larger networks than the rest of strategies.

Table 2 Centrality measures for the cluster heads

	$k=1$	$k=2$		$k=3$			$k=5$				
Cluster[Cluster Head ID]	C1 [27]	C1 [20]	C2 [39]	C1 [20]	C2 [39]	C3 [49]	C1 [20]	C2 [4]	C3 [49]	C4 [51]	C5 [39]
Cluster head degree	20	6	6	6	6	10	6	10	10	12	6
Neighborhood connectivity	7.7	9.6	9.6	9.6	9.6	10.8	9.6	8.7	10.8	8.1	9.6
Diameter	6	5	3	4	3	5	4	2	3	1	3
Random QMP - bandwidth [Mbps]	5.3	6.34		13.4			11.9				
Naive K-Means - bandwidth [Mbps]	16.6	18.3		23			23.4				
BASP - bandwidth [Mbps]	16.9	27.7		32.9			38.5				
BASP - running time [seconds]	46	28		17			9				

Regarding the comparison between *BASP* and *Random* placement, we find that the *Random* placement leads to an inefficient use of network's resources, and consequently to suboptimal performance. As depicted in the Fig. 11, the average gain of *BASP* over naive *Random* placement is 211% (i.e., $2x$ bandwidth gain).

Comparison to the Optimal Solution Note that our heuristic enables us to select cluster heads that provide much higher bandwidth than any other random or naive approach. But, if we were about to look for the optimum bandwidth within the clusters (i.e., optimum average bandwidth for the cluster), then this problem would be NP-hard. The reason is that finding the optimal solution entails running our algorithm for all the combinations of size k from a set of size n. This is a combinatorial problem that becomes intractable even for small sizes of k or n (e.g., $k = 5$, $n = 71$). For instance, if we wanted to find the optimum bandwidth for a cluster of size $k = 3$, then the algorithm would need to run for every possible (non-repeating) combination of size 3 from a set of 71 elements, i.e., $choose(71, 3) = 57K$ combinations. We managed to do so and found that the optimum average was 62.7 Mbps. For $k = 2$, the optimum was 49.1 Mbps. For $k = 1$, it was 16.9 Mbps.

The downside was that, the computation of the optimal solution took very long time in a commodity machine. Concretely, it took 5 hours for $k = 3$ and 30 minutes for $k = 2$. Instead, *BASP* spent only 17 seconds for $k = 3$ and 28 seconds for $k = 2$. Table 2 shows the improvement of *BASP* over *Random* and *Naive K-Means*. To summarize, BASP is able to achieve good bandwidth performance with very low computation complexity.

Correlation with Centrality Metrics Table 2 shows some centrality measures and some graph properties obtained for each cluster head. Further, Fig. 12 shows the neighborhood connectivity graph of the QMP network. The neighborhood connectivity of a node v is defined as the average connectivity of all neighbors of

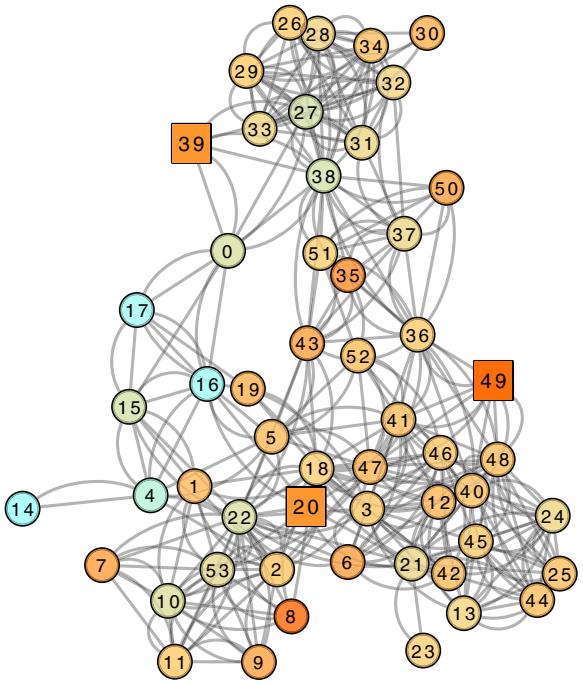


Fig. 12 Neighborhood connectivity graph of the QMP

v. In the figure, nodes with low neighborhood connectivity values are depicted with bright colors and high values with dark colors. It is interesting to note that some the nodes with the highest neighborhood connectivity are those chosen by *BASP* as cluster heads. The cluster heads (for $k = 2$ and $k = 3$) are illustrated with a rectangle in the graph. A deeper investigation into the relationship between service placement and network topological properties is out of the scope of this paper and will be reserved as our future work.

4.3 Dynamic Service Placement

In Guifi.net (i.e., QMP) nodes are added by the community members using their home's rooftop, which are often at non-optimal locations. This fact produces a high diversity in the quality of the links, making some nodes to be sporadically unreachable. Figure 13 depicts the number of nodes in the QMP network during the month of March 2017. Figure 13 reveals that there is a churn i.e., change in the set of participating nodes in the network, due to failures, electric cuts, nodes that have been upgraded, reconfigured, hanged, etc. The minimum number of nodes observed in the network is 67 and the maximum 74 nodes.

In order to see the performance of the BASP heuristic algorithm with churn of nodes, we run it in every day of March 2017. Figure 14 shows the average bandwidth to the cluster heads obtained with the naive K-Means and the BASP heuristic algorithm when using different number of services k ($k = 1$, $k = 2$, $k = 4$ and $k = 8$). Figure shows that the gap between the two algorithms grows as k increases. For instance when $k = 4$ (Fig. 14c), the average bandwidth to the cluster head obtained with K-Means algorithm is 18.9 Mbps and with BASP algorithm is 41 Mbps. This is because we keep clustering nodes by their bandwidth and the clusters are formed from the nodes with higher bandwidth.

Furthermore, we observed also some outliers in specific days of March 2017. For instance on 18th of March, Fig. 14a and b reveals a performance (i.e., bandwidth) drop. After performing some measurements we discovered that during these days one of the gateways (i.e., proxies) in the network got disconnected. Because of this, nodes that use this gateway to connect to the other nodes result in worst performance, since different paths are used (i.e., longer and slower). To summarize it, BASP outperforms the K-Means for every day of the month March and for the considered number of services k.

5 Experimental Evaluation

In order to foster the adoption and transition of the community micro-cloud environment, we provide a real community cloud distribution, codenamed

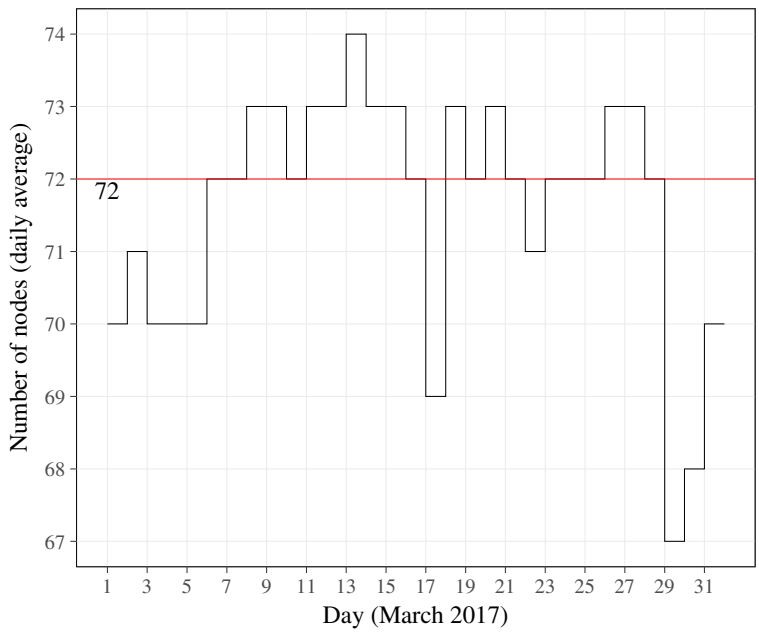
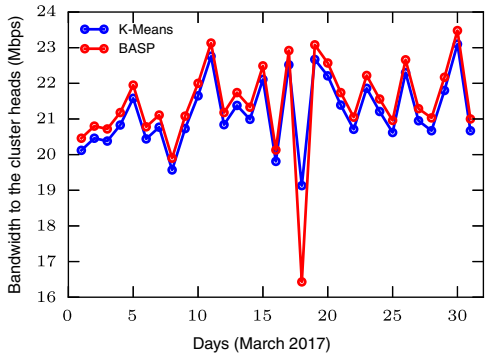
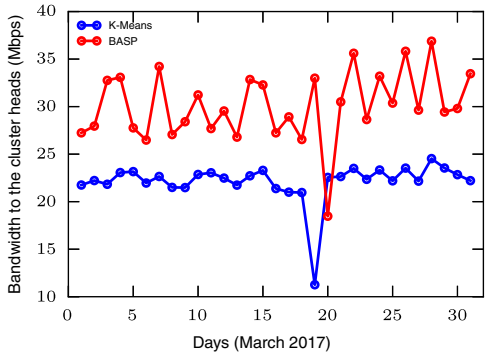


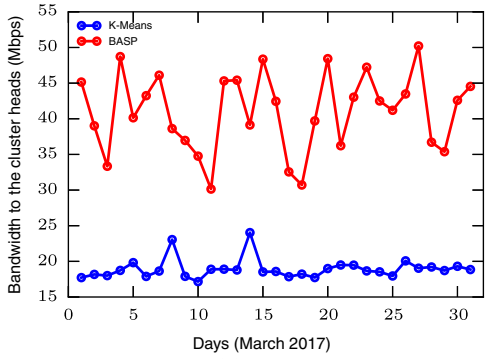
Fig. 13 Number of nodes in March 2017



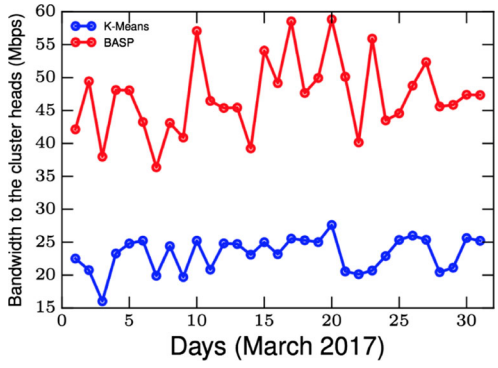
(a) $k = 1$



(b) $k = 2$



(c) $k = 4$



(d) $k = 8$

Fig. 14 K-Means vs. BASP (March 2017)

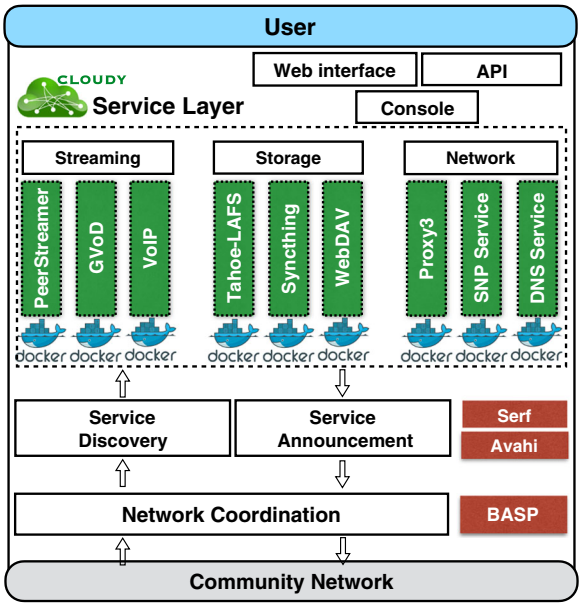


Fig. 15 Cloudy architecture

Cloudy [6], which contains the platform and application services of the community cloud system.

5.1 Cloudy: a Service Hub for the Micro-Clouds

Cloudy is the core software of our micro-clouds, because it unifies the different tools and services of the cloud system in a Debian-based Linux distribution. Cloudy is open-source and can be downloaded from public repositories.[7]

Cloudy's main components can be considered a layered stack, with services residing both inside the kernel and at the user level. Figure 15 reports some of the available services running on Docker containers. Cloudy includes a tool for users to announce and discover services in the micro-clouds based on Serf, which is a decentralized solution for cluster membership and orchestration. On the network coordination layer, having sufficient knowledge about the underlying network topology, the BASP decides about the placement of the service which then is announced via Serf as shown in Fig. 15. Thus, the service can be discovered by the other users.

[7] https://github.com/Clommunity/

5.2 Evaluation in a Real Production Community Network

In order to understand the gains of our network-aware service placement heuristic in a real production CN, we deploy our algorithm in real hardware connected to the nodes of the QMP network, located in the city of Barcelona. We concentrate on benchmarking two of the most popular network-intensive applications: *Live-video streaming service*, and *Web 2.0 service* performed by the most popular websites.

5.2.1 Live-Video Streaming Service

PeerStreamer,[8] an open source live P2P video streaming service, has been paradigmatically established as the live streaming service in Cloudy. This service is based on *chunk diffusion*, where peers offer a selection of the chunks they own to some peers in their neighborhood. A chunk consists of a part of the video to be streamed (i.e., by default, this is one frame of the video). PeerStreamer differentiates between a source node and a peer node. A source node is responsible for converting the video stream into chunks and sending to the peers in the network. In our case, both the source nodes and the peers run in Docker containers atop the QMP nodes.

Setup We use 20 real nodes connected to the wireless nodes of the QMP. These nodes are co-located in users homes (e.g., as home gateways, set-top-boxes, etc.). They run the Cloudy operating system. As the controller node, we leverage the experimental infrastructure of Community-Lab.[9] Community-Lab provides a central coordination entity that has knowledge about the network topology in real time and allows researchers to deploy experimental services and perform experiments in a production CN. The nodes of the QMP that are running the live video streaming service are part of the Community-Lab. In our experiments, we connect a live streaming camera (i.e., maximum bit-rate of 512 kbps, 30 frame-per-second) to a local PeerStreamer instance that acts as a source node.

The location of the source in such a dynamic network is therefore crucial. Placing the source in the QMP node with a weak connectivity will negatively impact the QoS and QoE of viewers. In order to determine the accuracy of the BASP upon choosing the appropriate QMP node where to host the source, we measure the average chunk loss percentage at the peer side, which is defined as the percentage of chunks that were lost and not arrived in time. This simple metric will help us understand the role of the network on the reliable operation of live-video streaming over a CN.

Our experiment is composed of 20 runs, where each run has 10 repetitions. Results are averaged over all the successful runs. Ninety percent of them were successful. In the 10% of failed runs, the source was unable to stream the captured images from the camera, so peers did not receive the data. This experiment was run for 2 weeks, with roughly 100 hours of live video data and several GBytes of logged content. The presented results are from one hour of continuous live streaming from the PeerStreamer source.

Results Figure 16 shows the average chunk loss for an increasing number of sources k. The data reveals that for any number of source nodes k, the BASP heuristic outperforms the currently adopted random placement in the QMP network. For $k = 1$, the BASP decreases the average chunk loss from 12 to 10%. This case corresponds to the scenario where there is one single source node streaming to the 20 peers in the QMP network. Based on the observations from Fig. 16, the gap between the two algorithms is growing as k increases. For instance, when $k = 3$, we get a 3% points of improvement with respect to chunk loss, and a significant 37% reduction in the packet loss rate.

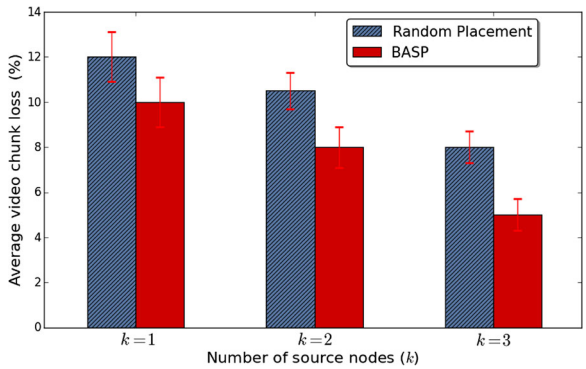


Fig. 16 Average video chunk loss

[8] http://peerstreamer.org/

[9] https://community-lab.net/

5.2.2 Web 2.0 Service

The second type of service that we experiment is the Web 2.0 Service. The workloads of Web2.0 websites differ from the workloads of older generation websites. Older generation websites typically served static content, while Web2.0 websites serve dynamic content. The content is dynamically generated from the actions of other users and from external sources, such as news feeds from other websites. We are experimenting with a social networking service, which is an example of a Microservices architecture, since it is formed by a group of independently deployable service components (i.e., web server, database server, memcached server and clients). In this type of service, the placement of the web server (together with the database server) is decisive for the user QoS.

Setup For the evaluation, we use the dockerized version of the CloudSuite Web Serving benchmark [26]. Cloudsuite benchmark has four tiers: the web server, the database server, the memcached server, and the clients. Each tier has its own Docker image. The web server runs Elgg[10] and it connects to the memcached server and the database server. The Elgg social networking engine is a Web2.0 application developed in PHP, similar in functionality to Facebook. The clients (implemented using the Faban workload generator) send requests to login to the social network and perform different operations. We use 10 available QMP nodes in total, where three of them act as a client. The other seven nodes are candidates for deploying the web server. The web server, database server and memcached server are always collocated in the same host. On the client side, we measure the response time when performing some operations such as login, live feed update, message sending, etc. In Cloudsuite, to each operation is assigned an individual QoS latency limit. If less than 95% of the operations meet the QoS latency limit, the benchmark is considered to be failed (i.e., marked as × in the Table 3). The location of the web server, database server and memcached server has a direct impact on the client response time.

Results Figure 17a and b depicts the response time observed by three clients for the update live feed operation, when placing the web server with the Random and the BASP, respectively.

Table 3 Cloudsuite benchmark results

Operations	Update live feed				Do login			
Threads	10	20	40	80	10	20	40	80
QMP-Random	✓	×	×	×	✓	✓	×	×
QMP-BASP	✓	✓	✓	×	✓	✓	✓	×
Stdev(sec)	0.02	0.03	0.01	0.01	0.02	0.02	0.01	0.03
Improvement	0.1	0.2	1.8	6.7	0.1	0.1	1.2	4.2

When placing the web server with the Random approach, Fig. 17a reveals that, as far as we increase the number of threads (i.e., concurrent operations) per client, the response time increases drastically in three clients. For up to 120 operations per client (i.e., 20 threads), all clients perceive a similar response times

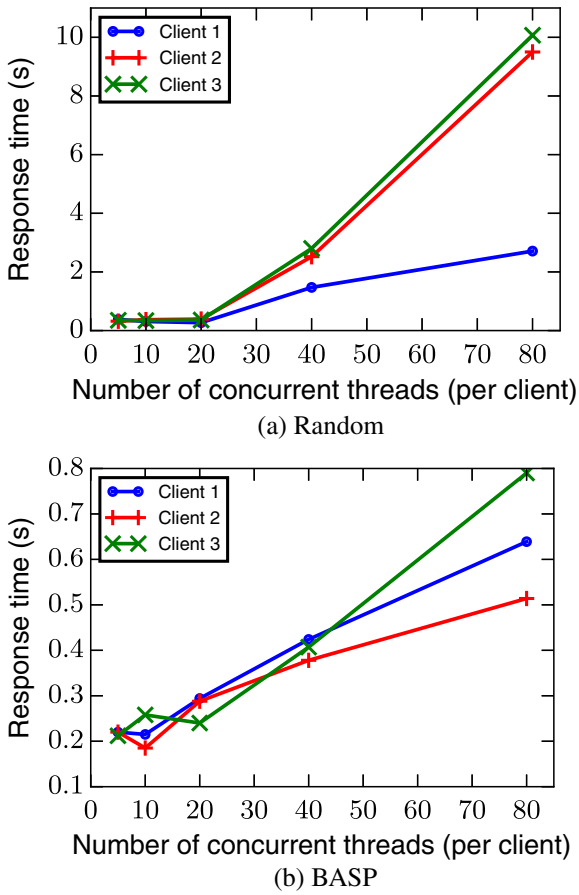


Fig. 17 Response times of clients (Random vs. BASP)

[10] https://elgg.org/

(300–350 ms). Response time increases more than one order of magnitude in Client 2 and Client 3, and an order of magnitude in Client 1 when performing 160 operations (i.e., 80 threads).

Figure 17b reveals that, the client response times for higher workloads *decreases an order of magnitude* when using our *BASP* heuristic compared to *Random* approach shown in the Fig. 17a (i.e., reaching 700 ms on average for 160 operations using BASP). For up to 120 operations per client, the response times that three clients perceive is slightly better (200–280 ms) than the response time when the web server is deployed with the *Random* approach. Furthermore, Table 3 demonstrates the successful and failed tests for the update feed and login operations in the Cloudsuite benchmark. Table reveals that, using the *BASP* heuristic the number of successful tests i.e., those that met the QoS latency limit, is higher than the number of successful tests with the *Random* approach. Further, it also shows the standard deviation values and average client response time improvements when using the *BASP* heuristic over *Random* approach. We can notice that the gain brought by the *BASP* heuristic is higher for more intensive workloads.

6 Related Work

Service placement is a key function of the cloud management systems. Typically, by monitoring all the physical and virtual resources on a system, service placement aims to balance load through the allocation, migration and replication of tasks. We looked at the service placement problem in four different environments: data center (DC), distributed data centers, wireless networks and IoT (Internet of things) environment.

Data Centers Choreo [19] is a measurement-based method for placing applications in the cloud infrastructures to minimize an objective function such as application completion time. Choreo makes fast measurements of cloud networks using packet trains as well as other methods, profiles application network demands using a machine-learning algorithm, and places applications using a greedy heuristic. Volley [2] is a system that performs automatic data placement across geographically distributed datacenters of Microsoft. Volley analyzes the logs or requests using an iterative optimization algorithm based on data access patterns and client locations, and outputs migration recommendations back to the cloud service. A large body of work of service placement in data centers has been devoted to finding heuristic solutions [16].

Most of the work in the data center environment is not applicable to our case because we have a strong heterogeneity given by the limited capacity of nodes and links, as well as asymmetric quality of wireless links. The difference/asymmetry in the link capacities across the network makes the service placement a very different problem than in a mostly homogeneous cloud datacenter. Our measurement results demonstrate that 25% of the links have a symmetry deviation higher than 40%.

Distributed Data Centers When the service placement algorithms decide how the communication between computation entities is routed in the substrate network, then we speak of network-aware service placement, i.e., closely tied to Virtual Network Embedding (VNE). The work in [36] proposes efficient algorithms for the placement of services in distributed cloud environment. The algorithms need input on the status of the network, computational resources and data resources which are matched to application requirements. In [18] authors propose a selection algorithm to allocate resources for service-oriented applications and the work in [3] focuses on resource allocation in distributed small datacenters. Another example of a network-aware approach is the work from Moens in [23] which employs a Service Oriented Architecture (SOA), where applications are constructed as a collection of services. Their approach performs node and link mapping simultaneously. The work in [34] extends the work of Moens in wireless settings taking into account the IoT. Mycocloud [15] is another work, which provides elasticity through self-organized service placement in decentralized clouds. The work of Elmroth [38] takes into account rapid user mobility and resource cost when placing applications in Mobile Cloud Networks (MCN). A recent work of Tantawi [37] uses biased statistical sampling methods for cloud workload placement. Regarding the service placement through migration, the authors in [39] and [46] study the dynamic service migration problem in mobile

edge-clouds that host cloud-based services at the n edge. They formulate a sequential decision making problem for service migration using the framework of Markov Decision Process (MDP) and illustrate the effectiveness of their approach by simulation using real-world mobility traces of taxis in San Francisco. The work in [22] evaluates the migration performance of various real applications in mobile edge clouds (MEC). The authors in [14] propose a fully approach to the joint optimization problem of scaling and placement of virtual network services. Spinnewyn [35] provides a resilient placement of mission-critical applications on geo-distributed clouds using heuristic based on subgraph isomorphism detection.

Most of the work in the distributed clouds consider micro-datacenters, where in our case the CN micro-clouds consist of constraint/low-power devices such us home gateways. Furthermore, in our case we have a partial information regarding the computational devices, so their approaches are not fully applicable to our environment.

Wireless Environment In [17] the authors propose an optimal allocation solution for ambient intelligence environments using tasks replication to avoid network performance degradation. Some other works done in wireless settings are the work of Vega [40] and our work [31] which proposes several placement algorithms that minimize the coordination and overlay cost along a CN. The work of Coimbra in [11] presents a parallel and distributed solution designed as a scalable alternative for the problem of service placement in CNs.

The focus of the work in this paper is to design a low-complexity service placement heuristic for CN micro-clouds in order to maximize bandwidth and improve user QoS and QoE.

IoT Environment The authors in [33] study the placement of IoT services on fog resources taking into account their QoS requirements. They show that their optimization model leads to 35% less cost of execution when compared to a purely cloud-based approach. Authors in [24] present a data placement strategy for Fog infrastructures called iFogStor. They formulate the data placement problem as a Generalized Assignment Problem (GAP) and propose heuristic one based on geographical zoning to reduce the solving time. Most of the IoT approaches analyzed are deployed in simulation environments using modeling and simulation toolkits such us iFogSim, thus, their results are not easily applicable to our context.

7 Conclusion

In this paper, we motivated the need for bandwidth and availability-aware service placement in CN micro-cloud infrastructures. CNs provide a perfect scenario to deploy and use community services in contributory manner. Previous work done in CNs has focused on better ways to design the network to avoid hot spots and bottlenecks, but did not relate to schemes for network-aware placement of service instances.

However, as services become more network-intensive, they can become bottlenecked by the network, even in well-provisioned clouds. In the case of CN micro-clouds, network awareness is even more critical due to the limited capacity of nodes and links, and an unpredictable network performance. Without a network-aware system for placing services, locations with poor network paths may be chosen while locations with faster, more reliable paths remain unused, resulting ultimately in a poor user experience.

We proposed a low-complexity service placement heuristic called BASP to maximize the bandwidth allocation when deploying CN micro-clouds. We presented algorithmic details, analyzed its complexity, and carefully evaluated its performance with realistic settings. Our experimental results show that the BASP consistently outperforms the currently adopted random placement in Guifi.net by $2x$ bandwidth gain. Moreover, as the number of services increases, the gain tends to increase accordingly. Furthermore, we deployed our service placement algorithm in a real network segment of the QMP network, a production CN, and quantified the performance and effects of our algorithm. We conducted our study on the case of a live video streaming service and Web 2.0 Service integrated through Cloudy distribution. Our real experimental results show that when using BASP heuristic algorithm, the video chunk loss in the peer side is decreased up to 3% points, i.e., worth a 37% reduction in the packet loss rate. When using the BASP with the Web 2.0 service, the client response times decreased up to an order of magnitude, which is a significant improvement.

As a future work, we plan to look into live service migration, i.e., the controller needs to decide which micro-cloud should perform the computation for a particular user, with the presence of user mobility and other dynamic changes in the network.

Acknowledgements This work was supported by the European H2020 framework program projects RIFE (H2020-644663), netCommons (H2020-688768), LightKone (H2020-732505), and by the Spanish government under contract TIN2016-77836-C2-2-R. This work was also supported by the national funds through Fundação para a Ciência e a Tecnologia in project ContexTWA with reference PTDC/EEI-SCR/6945/2014.

References

1. Stirling Number of the Second Kind. http://mathworld.wolfram.com/StirlingNumberoftheSecondKind.html
2. Agarwal, S., et al.: Volley: automated data placement for geo-distributed cloud services. In: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, NSDI'10, pp. 2–2. USENIX Association, Berkeley (2010)
3. Alicherry, M., Lakshman, T.V.: Network aware resource allocation in distributed clouds. In: Proceedings of INFOCOM, IEEE, pp. 963–971 (2012)
4. Apolónia, N., Freitag, F., Navarro, L.: Leveraging deployment models on low-resource devices for cloud services in community networks. Simul. Model. Pract. Theory **77**, 390–406 (2016)
5. Baig, R., Dalmau, L., Roca, R., Navarro, L., Freitag, F., Sathiaseelan, A.: Making Community Networks Economically Sustainable, the Guifi.Net Experience. GAIA '16, pp. 31–36. ACM, New York (2016)
6. Baig, R., Freitag, F., Navarro, L.: Cloudy in guifi.net: establishing and sustaining a community cloud as open commons. Futur. Gener. Comput. Syst. (2018)
7. Baig, R., Roca, R., Freitag, F., Navarro, L.: guifi.net, a crowdsourced network infrastructure held in common. Comput. Netw. **90**, 150–165 (2015). Crowdsourcing
8. Bilalli, B., Abelló, A., Aluja-Banet, T., Wrembel, R.: Intelligent assistance for data pre-processing. Computer Standards & Interfaces **57**, 101–109 (2018)
9. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12, pp. 13–16. ACM, New York (2012)
10. Cerdà-Alabern, L., Neumann, A., Escrich, P.: Experimental evaluation of a wireless community mesh network. In: Proceedings of the 16th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM '13, pp. 23–30. ACM, New York (2013)
11. Coimbra, M.E., Selimi, M., Francisco, A.P., Freitag, F., Veiga, L.: Gelly-scheduling distributed graph processing for service placement in community networks. In: 33rd ACM/SIGAPP Symposium on Applied Computing (SAC 2018). ACM (2018)
12. Dimogerontakis, E., Meseguer, R., Navarro, L.: Internet Access for All: Assessing a Crowdsourced Web Proxy Service in a Community Network, pp. 72–84. Springer International Publishing, Cham (2017)
13. Dimogerontakis, E., Neto, J., Meseguer, R., Navarro, L.: Client-side routing-agnostic gateway selection for heterogeneous wireless mesh networks. In: IFIP/IEEE International Symposium on Integrated Network Management (IM) (2017)
14. Draxler, S., Karl, H., Mann, Z.A.: Joint optimization of scaling and placement of virtual network services. In: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 365–370 (2017)
15. Dubois, D.J., Valetto, G., Lucia, D., Di Nitto, E.: Mycocloud: elasticity through self-organized service placement in decentralized clouds. In: 2015 IEEE 8th International Conference on Cloud Computing, pp. 629–636 (2015)
16. Ghanbari, H., et al.: Replica placement in cloud through simple stochastic model predictive control. In: 2014 IEEE 7th International Conference on Cloud Computing, pp. 80–87 (2014)
17. Herrmann, K.: Self-organized service placement in ambient intelligence environments. ACM Trans. Auton. Adapt. Syst. **5**(2), 6:1–6:39 (2010)
18. Klein, A., Ishikawa, F., Honiden, S.: Towards network-aware service composition in the cloud. In: Proceedings of the 21st International Conference on World Wide Web, WWW '12, pp. 959–968. ACM, New York (2012)
19. LaCurts, K., et al.: Choreo: network-aware task placement for cloud applications. In: Proceedings of the 2013 Conference on Internet Measurement Conference, IMC '13, pp. 191–204. ACM, New York (2013)
20. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. Phys. Rev. E **80**, 056117 (2009)
21. Lertsinsrubtavee, A., Ali, A., Molina-Jimenez, C., Sathiaseelan, A., Crowcroft, J.: Picasso: a lightweight edge computing platform. In: IEEE 6th International Conference on Cloud Networking (Cloudnet'17) (2017)
22. Machen, A., Wang, S., Leung, K.K., Ko, B.J., Salonidis, T.: Live service migration in mobile edge clouds. In: IEEE Wireless Communications (2017)
23. Moens, H., et al.: Hierarchical network-aware placement of service oriented applications in clouds. In: 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1–8 (2014)
24. Naas, M.I., Raipin, P., Boukhobza, J., Lemarchand, L.: iFogStor: an IoT data placement strategy for fog infrastructure. In: IEEE 1st International Conference on Fog and Edge Computing. Madrid, Spain (2017)

25. Neumann, A., Lopez, E., Navarro, L.: An evaluation of Bmx6 for community wireless networks. In: 8th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (Wimob), 2012 I, pp. 651–658 (2012)
26. Palit, T., Shen, Y., Ferdman, M.: Demystifying cloud benchmarking. In: 2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 122–132 (2016)
27. Selimi, M., Cerdà-Alabern, L., Sánchez-Artigas, M., Freitag, F., Veiga, L.: Practical service placement approach for microservices architecture. In: Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid '17, pp. 401–410. IEEE Press, Piscataway (2017)
28. Selimi, M., et al.: Integration of an assisted P2p live streaming service in community network clouds. In: Proceedings of the IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom 2015). IEEE (2015)
29. Selimi, M., Freitag, F., Cerdà-Alabern, L., Veiga, L.: Performance evaluation of a distributed storage service in community network clouds. Concurrency and Computation: Practice and Experience **28**(11), 3131–3148 (2016). cpe.3658
30. Selimi, M., Khan, A.M., Dimogerontakis, E., Freitag, F., Centelles, R.P.: Cloud services in the guifi.net community network. Comput. Netw. **93**, Part 2:373–388 (2015)
31. Selimi, M., Vega, D., Freitag, F., Veiga, L.: Towards Network-Aware Service Placement in Community Network Micro-Clouds, pp. 376–388. Springer International Publishing, Berlin (2016)
32. Sharifi, L., Cerdà-Alabern, L., Freitag, F., Veiga, L.: Energy efficient cloud service provisioning: keeping data center granularity in perspective. Journal of Grid Computing **14**(2), 299–325 (2016)
33. Skarlat, O., Nardelli, M., Schulte, S., Dustdar, S.: Towards Qos-aware fog service placement. In: IEEE International Conference on Fog and Edge Computing (ICFEC 2017). Madrid, Spain (2017)
34. Spinnewyn, B., Braem, B., Latré, S.: Fault-tolerant application placement in heterogeneous cloud environments. In: Network and Service Management (CNSM), pp. 192–200 (2015)
35. Spinnewyn, B., Mennes, R., Botero, J.F., Latré, S.: Resilient application placement for geo-distributed cloud networks. J. Netw. Comput. Appl. **85**, 14–31 (2017). Intelligent Systems for Heterogeneous Networks
36. Steiner, M., et al.: Network-aware service placement in a distributed cloud environment. In: Proceedings of the ACM SIGCOMM 2012 Conference, SIGCOMM '12, pp. 73–74. ACM, New York (2012)
37. Tantawi, A.N.: Solution biasing for optimized cloud workload placement. In: 2016 IEEE International Conference on Autonomic Computing (ICAC), pp. 105–110 (2016)
38. Tarneberg, W., Mehta, A., Wadbro, E., Tordsson, J., Eker, J., Kihl, M., Elmroth, E.: Dynamic application placement in the mobile cloud network. Futur. Gener. Comput. Syst. **70**, 163–177 (2017)
39. Urgaonkar, R., Wang, S., He, T., Zafer, M., Chan, K., Leung, K.K.: Dynamic service migration and workload scheduling in edge-clouds. Perform. Eval. **91**, 205–228 (2015)
40. Vega, D., Meseguer, R., Cabrera, G., Marques, J.M.: Exploring local service allocation in community networks. In: 10th International Conference on Wireless and Mobile Computing, Networking and Communications (Wimob'14), IEEE, pp. 273–280 (2014)
41. Vega, D., Baig, R., Cerdà-Alabern, L., Medina, E., Meseguer, R., Navarro, L.: A technological overview of the guifi.net community network. Comput. Netw. **93**, Part 2:260–278 (2015)
42. Vega, D., Cerdà-Alabern, L., Navarro, L., Meseguer, R.: Topology patterns of a community network: Guifi.net. In: 1st International Workshop on Community Networks and Bottom-Up-Broadband (CNBub 2012), within IEEE Wimob. Barcelona, Spain, pp. 612–619 (2012)
43. Verespej, H., Pasquale, J.: A characterization of node uptime distributions in the Planetlab test bed. In: 2011 IEEE 30th International Symposium on Reliable Distributed Systems, pp. 203–208 (2011)
44. Wang, L., Bayhan, S., Ott, J., Kangasharju, J., Sathiaseelan, A., Crowcroft, J.: Pro-diluvian: understanding scoped-flooding for content discovery in information-centric networking, pp. 9–18. ACM, New York (2015)
45. Wang, S., Zafer, M., Leung, K.K.: Online placement of multi-component applications in edge computing environments. IEEE ACCESS **5**, 2514–2533 (2017)
46. Wang, S., Urgaonkar, R., He, T., Chan, K., Zafer, M., Leung, K.K.: Dynamic service placement for mobile micro-clouds with predicted future costs. IEEE Trans. Parallel Distrib. Syst. **28**(4), 1002–1016 (2017)